X.commerce Workshop – PHP

Setup and Configuration

This lab assumes an intermediate level of experience with PHP as a programming language, and also assumes that you are running an Apache 2 web server to serve your PHP pages.

Also, any shell commands assume that you're working in a flavor of Unix (including Cygwin), but they should be easily translated into DOS commands, if you're working in Windows.

Beyond Apache2 and PHP5, all of the software that you need to complete this lab is provided in the X.commerce Developer Package, which you should have already installed, and the supplementary xdevcamp-php.zip file. Before we get started, please unzip the contents of the xdevcamp-php.zip file into your directory of choice, e.g. C:\xdevcamp.

Lab 1: Setup the Fabric

Setup Mongo

1. Create Mongo data directory: **`mkdir /data/db`**
2. Start MongoDB: **`~/dev/mongodb-osx-x86_64-1.8.2/bin/mongod &`**

Start the Fabric

1. Using Ant: From the root of the developer package, run: **`ant xfabric.start`**
2. Without Ant: **`java -jar main-xfabric-0.5.4.jar`**

Open the Manager UI

1. In your favorite modern web browser, open: http://localhost:8079/
2. Login with default credentials: **`admin`** / **`password0`**

Lab 2: Run AuctionHouse application

Create Auction Topics

1. Open the the Manager UI
2. Click the "Topics" tab
3. Scroll down to the "Define Topic" form
4. Create the following topics:

Topic Name	Tenant Require?

/auction/ended	No
/auction/started	No
/auction/updated	No
/bid/accepted	Yes
/bid/extended	Yes
/bid/outbid	Yes
/bid/placed	Yes
/bid/rejected	Yes
/bid/won	Yes

Define Capability in the Manager

1. In the Capabilities drop-down, select "Sample Capability 1"
2. Enter **`http://localhost:8888/`** in the "Endpoint" field and then click "Update".

Subscribe Capability to Topics

1. Click on the "Subscriptions" tab
2. Enter **`/bid/placed`** in the "Topic Pattern" field and click "Subscribe"

Register Tenants with the Capability

1. Click on the "Authorization Info" tab
2. Ensure that "merchant 2" and "merchant 3" are in the "Authorizations" table.
3. If not, add them with the "Authorize Tenant" form.

Start AuctionHouse application

1. On the command-line, navigate to the folder where you unzipped the xdevcamp-auction-java.zip file.
2. Run
   ```
   java –jar common/auctionhouse-0.1-bundle.jar
   http://localhost:8080/ 8888
   ```

See Messages

1. Back in the Manager UI, click on the "Message Tracing" tab
2. In the "Find Recent Messages" form, click "Search"
3. You will see a list of message IDs, one for each message that the AuctionHouse app has published.

Lab 3: Create new capability

Create a folder for the new project

Locate web root

1. Locate and open your `httpd.conf` file
2. Find the "DocumentRoot" directive and make note of the configured root document folder, e.g. `/var/www`
3. Exit back to the shell
4. Make a "bidder" subdirectory of the root document folder:
   `mkdir /var/www/bidder`

Add Avro library

Copy the files from <assets-dir>/php/avro to /var/www/bidder.

Test the setup

1. In the “bidder” directory, create a “test.php” file.
2. Enter the following contents in the file:

```
<?php  phpinfo() ?>
```

3. Open the file in your browser: http://localhost/bidder/test.php
4. You will see a diagnostic HTML page that lists your PHP version and system settings.

Lab 4: Register capability with the Fabric

Register new capability with the Fabric

1. Open the Manager UI: http://localhost:8079/
2. In the Capabilities drop-down, select the "Sample Capability 2" capability
3. Set the Endpoint field to **`http://localhost/bidder`** and then click "Update"

Subscribe Capability to Topics

1. Click on the "Subscriptions" tab
2. Enter the following topics in the “Topic Pattern” field and click "Subscribe":
   - **`/auction/started`**
   - **`/auction/updated`**
   - **`/auction/ended`**
   - **`/bid/accepted`**
   - **`/bid/extended`**
   - **`/bid/outbid`**
   - **`/bid/rejected`**
   - **`/bid/won`**

Register Tenants

1. Click on the "Authorizations" tab
2. Ensure that “merchant 2” and “merchant 3” are in the “Authorizations” table.
3. If not, add them with the “Authorize Tenant” form.

Validate that messages are being delivered

1. Click on the "Message Trace" tab
2. In the “Find Recent Messages” form, select capability "Sample Capability 2" and click "Search"
3. Click on one of the resulting messages to see details
   Note: Getting a 404 message status is expected because the server you are sending a message to doesn’t know how to process it yet.

Lab 5: Start Receiving Messages

Create AuctionConsole script

1. In the <root-dir>/bidder folder, create a "AuctionConsole.php" file.
2. Add code to import the Avro library:
```
<?php
include "avro/avro.php";
?>
```
3. Add helper function to get HTTP header values:
```
function getHeaderValue($headerName) {
    $headers = getallheaders();
    $value = NULL;
    if (array_key_exists($headerName, $headers)) {
        $value = $headers[$headerName];
    }
    return $value;
}
```
4. Add helper function to decode Avro messages into an associative array:
```
function getMessageObject($data) {
    $read_io = new AvroStringIO($data);
    $data_reader = new AvroDataIOReader($read_io, new
AvroIODatumReader());

    $results = array();
    foreach ($data_reader->data() as $datum) {
      array_push($results, $datum);
    }
    return $results;
}
```
5. Add code to authenticate incoming messages:
```
if (getHeaderValue("Authorization") != "Bearer
QlVTRk9SQVVUSElELTEA7/GRzPeAKiymgVsONHEikg==") {
    header("HTTP/1.0 401 Unauthorized");
    exit();
}
$tenant = getHeaderValue("X-XC-TENANT-ID");
```
6. Add code to track the message ID:
```
$messageGuid = getHeaderValue("X-XC-MESSAGE-GUID");
```
7. Add code to deserialize the POST data into an array:
```
$post_data = file_get_contents("php://input");
$messages = getMessageObject($post_data);
```

8. Add code to process the message based on the topic

```
$topic = substr($_SERVER['REQUEST_URI'], strlen("/bidder"));
$logMessage = "[$messageGuid][$tenant] $topic:";
if (strpos($topic, "/auction/") === 0) {
        $listing = $messages[0];
        $logMessage .= $listing["xId"].":".$listing["title"]." --
$".$listing["price"]["amount"];
} else if (strpos($topic, "/bid/") === 0) {
        $bid = $messages[0];
        $logMessage .= $bid["listingId"]." ->
$".$bid["bidAmount"]["amount"];
} else {
        $logMessage .= "Rogue Message";
}
file_put_contents("auction_log", $logMessage."\n", FILE_APPEND);
```

Setup log file

1. From the command line, create the log file: **`touch auction_log`**
2. Make the file writable: **`chmod a+w auction_log`**

Add Apache config to route messages

1. In the <root-dir>/bidder folder, create a .htaccess file
2. Add the following lines:

```
RewriteEngine   On
RewriteBase     /bidder
RewriteRule     ^[^\.]*$        AuctionConsole.php
```

3. Restart Apache, e.g.: **`httpd -k restart`**

View incoming messages

1. Run "tail" to view the data in auction_log: **`tail -f auction_log`**
2. You will start seeing messages like the following:
   `[9b7fe54a-c9d6-422c-b7fe-e2decdf4c788][]`
   `/auction/updated:106:Size 3 Pink Soccer Ball -- $0`
   These will get more interesting after the next lab.

Lab 6: Start Publishing Messages

Update Apache config

In the .htaccess file, add a RewriteRule to send requests for /bidder/makeBid to your Bidder.php:

```
RewriteEngine   On
RewriteBase /bidder
RewriteRule     ^makeBid$ Bidder.php    [L]
RewriteRule ^[^\.]*$    AuctionConsole.php
```

Create Bidder script

1. Copy Auction.avpr and Marketplace.avpr files to the <root-dir>/bidder folder
2. In the <root-dir>/bidder folder, create a "Bidder.php" file
3. In the Bidder.php file, add code to import the Avro library:

```
<?php
```

```
include "avro/avro.php";
?>
```

4. Add helper function to send an HTTP request:

```
function http_request($ch)
{
    $response = curl_exec($ch);
    $error = curl_error($ch);
    $result = array( 'header' => '',
                     'body' => '',
                     'curl_error' => '',
                     'http_code' => '',
                     'last_url' => '');
    if ( $error != "" )
    {
        $result['curl_error'] = $error;
        return $result;
    }

    $header_size = curl_getinfo($ch,CURLINFO_HEADER_SIZE);
    $header_block = substr($response, 0, $header_size);
    $allheaders = explode("\r\n", $header_block); // split the header
text into lines
    array_shift($allheaders); // drop the first line since that's the
HTTP status line
    $headers = array();
    foreach($allheaders as $header) {
        $splitheader = explode(": ", $header, 2);
        if (count($splitheader) == 2) {
            $headers[$splitheader[0]] = $splitheader[1];
        }
    }
    $result['headers'] = $headers;
    $result['body'] = substr( $response, $header_size );
    $result['http_code'] = curl_getinfo($ch,CURLINFO_HTTP_CODE);
    $result['last_url'] = curl_getinfo($ch,CURLINFO_EFFECTIVE_URL);

    if ($result['http_code'] != 200) {
        $result['curl_error'] = $result['http_code'];
    }

    return $result;
}
```

5. After the "?>" character, add the form that will let you submit a bid:

```
<html>
<body>
<form method="POST">
<label>Bidder: </label><input name="tenant" type="text" /><br/>
<label>Listing ID: </label><input name="listingId" type="text" /><br/>
<label>Bid Price: </label> $<input name="bid" type="text" /><br/>
<button type="submit">Post Bid</button>
</form>
</body>
</html>
```

6. Add the following code within the <?php ?> block to collect information for the bid:

```
if  ($_POST) {
    $tenant = $_POST["tenant"];
    $listingId = $_POST["listingId"];
    $bidAmount = floatval($_POST["bid"]);
}
```

7. In side the post block, create a bid based on the provided data:

```
$bid = array('listingId'  => $listingId,
             'bidAmount'  => array('amount'  => $bidAmount,
                                   'code'  => 'USD'));
```

8. Encode the bid using Avro:

```
$protocol = AvroProtocol::parse(file_get_contents("Auction.avpr"));
$schemata = $protocol->schemata;
$schema = new
AvroUnionSchema(array("com.x.service.marketplace.message.CurrencyAmount",
"com.x.devcamp.auction.Bid"),
            "com.x.devcamp.auction", $schemata, TRUE);
$strio = new AvroStringIO();
$dw = new AvroDataIOWriter($strio, new AvroIODatumWriter($schema),
$schema);
$dw->append($bid);
$dw->close();
$messageData = $strio->string();
```

9. Convert tenant to tenant auth token

```
$tenantAuth = "bidder1"  == $tenant ?
"QVVUSElELTEAuRyP8LTHzE/oooUzVdZZdQ=="  :
"QVVUSElELTEAS9f0EZZjuvH+w7wQP/KM1A==";
```

10. Publish bid message

```
$ch = curl_init();
curl_setopt($ch, CURLOPT_URL, 'http://localhost:8080/bid/placed');
curl_setopt($ch, CURLOPT_HEADER, true);
curl_setopt($ch, CURLOPT_RETURNTRANSFER, true);
curl_setopt($ch, CURLOPT_TIMEOUT, 10);
curl_setopt($ch, CURLOPT_POST, TRUE);
curl_setopt($ch, CURLOPT_POSTFIELDS, $messageData);
curl_setopt($ch, CURLOPT_HTTPHEADER, array("Authorization: Bearer
$tenantAuth", "Content-Type: avro/binary"));
$output = http_request($ch);
curl_close($ch);
```

11. Add some logging to handle errors and display success:

```
if  ($output['curl_error'] != '') {
    echo "<div class='error'>".$output['curl_error']."</div>";
} else {
    echo "<div>Message ".$output['headers']['X-XC-MESSAGE-GUID']."
Sent</div>";
}
```

Make an initial bid

1. Open web browser to http://localhost/bidder/makeBid
2. Fill in the form with the following values (replace '114' with whatever listing ID is currently on the block in your auction_log):

```
Bidder: bidder1
Listing ID: 114
Bid Price: 5
```

3. The auction_log will show the accepted bid

Make a subsequent bid

1. Fill in the form with the following values:

```
Bidder: bidder2
Listing ID: 114
Bid Price: 10
```

2. The auction_log will show the bid accepted and outbid messages